CS206

Midterm

Iterators

Recursion
Midterm

- Q1 — reading code
  - 16 pts — mean 14.2
- Q2 — copy into array list
  - 20 pts — mean 17.1
- Q3 — Big-O
  - 16 pts — mean 13.4
- Q4 — bubble swap on linked lists
  - 24 pts — mean 15.9
- Q5 — queue copying
  - 24 pts — mean 18.9
- Overall mean 80.3
Q1

What is the output?

A. PartA.java
B. Overloaded.java
C. SNums.java
D. AlFunc.java
Write a method that takes as input two arrays of Strings and copies only those strings that occur at the same index in both arrays to a new ArrayList, which is returned.

copier.java
Q3

Complexity

TD.java
Q4
BubbleSwap on a doubly linked list

BubbleSwapList.java
Q5

Merge 2 queues into one, keeping sorted order

ArrayQueue.java  merge function
Iterators

- Abstracts the process of scanning through a sequence of elements (traversal)
  - an interface with three methods
    - boolean hasNext()
      - true if the iteration has more elements
    - E next()
      - Returns the next element
    - void remove()
      - Removes from the underlying collection the last element (optional)

- Combination of these two methods allow a general traversal structure
  ```java
  while(iter.hasNext()) {
      iter.next();
  }
  ```
Why Iterators?

- They encapsulate traversal
- Container independence
  - allows traversal without knowledge of underlying data structure implementation, i.e. `.length` or `.size()`
  - allows switching out the underlying data structure while causing the least amount of code change elsewhere
Iterable Interface

• An interface that with a single method:
  - `iterator()` returns an iterator of the elements in the collection

• Each call to `iterator()` returns a new iterator instance, thereby allowing multiple independent traversals of a collection
Iterator example

IteratorTest.java

1 — no iterator, dies on remove
2 — no iterator, unexpected behavior
3 — iterator, success
Writing to Files

- In the simplest case as easy as println
  - outputter.java

- Lots for for complex scenarios
  - java.io
  - java.nio.channel
  - java.nio.files
Recursion

A method that calls itself, either directly or indirectly
Importantly, need a way to stop

```java
public void a(int c)
{
    System.out.println("A" + c);
    a(c-1);
}
```

```java
public void b(int c)
{
    System.out.println("B" + c);
    if (c<=0) return;
    b(c-1);
}
```

Class Recurser